Software engineers face fundamental challenges

Galen Gruman, Assistant Editor
The world’s developed nations are becoming information societies that will require programmers and software engineers to resolve fundamental issues in productivity and advance key technologies to be successful, said speakers at the 11th Annual Computer Software and Applications Conference held Oct. 5-9 in Tokyo.

According to conference presentations, the key issues for software developers to address are:
- How to manage software development,
- How to assure software quality,
- How to develop and apply powerful database architectures and distributed processing,
- How to integrate computer technology with people’s needs through expert systems, language processing (natural and computer), data-interchange protocols, and operating systems designed for everyday users, and
- How to cope with a growing shortage of top-level programmers and software engineers caused by demand for software that is increasing twice as fast as the rate of new programmers (by 2000, Japanese government figures predict a shortfall of 970,000 programmers in Japan).

Answers did not exist for most of these issues, but participants did agree that researchers in the US, Europe, and Japan have recognized the fundamental problems now facing the software industry and have begun several comprehensive projects to investigate solutions.

“We weren’t sure what to do 10 years ago, but today we know the direction we are going in,” said Toshirō Ohno of the Japan Business Automation Co. at a tutorial session on the major technology effects on software engineering.

One widely held view was that software development must be treated systematically, not as one-time, ad hoc projects. “Software is [now] tailored to the individual needs of a customer, not to the general needs of the software market,” said Akira Nagashima, director of microprocessor-based scientific tools for the Sigma software-development environment project, at the tutorial session. That must change, he said.

“We who are engaged in the production of software are concerned that, although it is invisible, it is a commodity that must be sold like hardware,” said Tadahiro Sekimoto, president of NEC Corp. and one of Compsac’s three keynote speakers. Software has development, testing, sales, and postsales service phases just like hardware, he told the 683 conference attendees.

“In the area of software, it is very important to recognize that productivity is very important,” Sekimoto said. To increase productivity, “we have to rationalize it,” he said. “The things software can do will be the critical paths. [But its] limits will be the limits on quality of life,” he said. “Those who will be able to conquer software will be able to conquer the world,” Sekimoto said.

Information flow. Software technology must be transferred equitably among software-producing nations, said L. Desaix Anderson, the US chargé d’affaires in Japan, also at the keynote session. He called on Japan to accept the US and European format of intellectual property rights (such as copyrights), saying that “equitable transfer encourages more development . . . [while] a pattern of disputes can slow down progress.” The information must flow from Japan, not just to it, he said.

The US and Japan are negotiating an umbrella science and technology agreement to equalize the flow of information between them, said William McPherson, a US Embassy second secretary specializing in science and technology policy, in an interview after the conference. “We’re very interested in what they’re doing in the government labs and in some private labs,” he said.

The embassy has looked at two Japanese information services — the Japan Information Center for Science and Technology and the National Academic Center for Science Information Systems — that have some English indexes and abstracts. Of the several million items in the Japan Information Center, about a half million have English index information, and 1300 English entries are being added each month, McPherson said. The service is available in the US as well as in Japan. Japan has offered to give the National Science Foundation in Washington, DC, a terminal to access the other service, he said.

Japan, McPherson said, should be more forthcoming in sharing its industry-generated research and make its intellectual-property laws protect the ideas’ originators longer and better. But he acknowledged that US researchers must also make an effort to get information from Japan. Very few Americans (between 100 and 150 last year) come to Japanese firms under exchange programs, and those who do stay for less than two months, he said.

Japan, too, has opened doors to foreigners that were traditionally closed, McPherson said. Japan’s Science and Technology Agency, for example, has recently begun a program to hire foreigners. Under the leadership of Prime Minister Yasuhiro Nakasone, Japan has unilaterally opened some doors, McPherson said. Nakasone’s term ended Oct. 20, but the three candidates to replace him have said they will continue his door-opening policies. However, the programs that have begun have very small budgets, usually less than a million dollars, McPherson said.

Continued on p. 96
CASE: How real is it?

Janet Butler, Applied Computer Research

"The future of computing is a battle with complexity," argued James Martin, a Pulitzer Prize finalist and author of more than 50 books on computers and communications technology, at the ShowCASE conference sponsored by Washington University and held Sept. 1-2 in St. Louis. "Systems like SDI [Strategic Defense Initiative] are enormously complex and can't be built manually. We need very fast and efficient software development. [But] we're still writing software with pencils and plastic templates," he said. "Programmers need to catch up with engineers and use software tools."

Ken Orr, a leading CASE consultant, said, "We haven't automated the software-development process before because we didn't know how. We could do it intellectually, but it wasn't converted into procedures." But it must be done, he said, "because it's a problem that has to be solved. There are too few professionals capable of building the systems we need. Problems of size and complexity are driving us to use CASE."

But, Orr said, CASE threatens middle managers, good technicians who don't see themselves as managers and who don't know how to manage the new technology. Vaughn Merlyn, an application-development automation consultant, agreed. While software-engineering tools may be bought for one project, they are rarely used on others. "There's a failure to institutionalize the technology," he said.

If CASE is to be the systems-development environment of the future, what will it be like and what will it cost? capers Jones, a software-productivity expert, described his vision of a system in 1995. The configuration would include mainframe databases, minicomputer file servers, primary workstations, portable workstations, home computers, and documentation computers. This system would have more than 100 integrated tools and more than 20 expert systems, plus standard data-interchange and data-conversion utilities. The CASE software would be partitioned into management, technical-support, and physical-environments areas. He estimated it would cost $83.5 million.

What benefits would such an expensive system offer? Jones said for major systems it would give a less-than-10-percent cancellation rate (versus today's 25 percent rate), a one- to two-year development cycle (versus today's three- to 10-year cycle), and million-dollar budgets (versus today's $ multimillion-dollar budgets). Systems produced from this system would be reliable, have high quality, and require little maintenance, he said.

Butler is managing editor of the System Development newsletter, whose November issue focuses on CASE (ACR, PO Box 9280, Phoenix, AZ 85068; $15 single copy, $120 subscription).

---

NEWS BRIEFS

Boston University boosts SE program. After acquiring the defunct Wang Institute earlier this year, Boston University has expanded its software-engineering masters degree program. The program is part of the university's masters of science program in systems engineering. The new program will begin in January, said John Brackett, a former Wang Institute professor now teaching software engineering at Boston University.

The program will have two curricula: one for students with a hardware background and one for those with a software background. The two curricula are identical except that the hardware track requires a data-structures course and a operating systems course while the software track requires a switching-theory course and a computer architecture course. Both curricula will emphasize software engineering for embedded systems and networked systems, Brackett said.

Because of industry demand for such a program, Brackett has established a corporate associates program so local companies can help pay for the program they said they wanted to keep, he said. Associate companies are guaranteed annual admission slots, the number of which is based on their membership fees.

Programmer jobs continue to grow. The number of programmers and software developers in the US grew 8.6 percent to 218,500 from July 1986 to July 1987, according to US Labor Dept. statistics. That pace surpasses the 6.2-percent growth seen between November 1983 and November 1986, when the 1985 computer-industry slump apparently bottomed out. The overall computing and data-processing field saw a 7.5-percent rise in employment to 640,900 from July 1986 to July 1987.

In the same period (the latest available), wages increased by 2.8 percent to an average of $454.96 a week for the overall computing field. (The department does not track programmer wages.) The amount of hours worked each week declined 1.5 percent to 37.6 hours a week.
Space station hands NASA its biggest software task yet

John Wolfsberger, NASA
Marshall Space Flight Center

To create a permanent, manned space station, the National Aeronautics and Space Administration will divide the development effort among four centers, requiring the agency to manage a distributed project larger than anything it has ever attempted to handle. NASA engineers must integrate and test the software systems from several independent, widely separated designers and perform long-term activities over as many as 30 years.

The project's size and complexity prompted a NASA software working group to recommend that:
- A consistent software management approach be developed and applied to all software-contributing participants.
- Software standards and policies be established, implemented, and enforced throughout the program.
- NASA establish a common software environment that everyone uses to produce customized software.
- One language — Ada — be designated for all space station application software.

NASA headquarters has since implemented the recommendations by letting a contract for a technical and management information system (TMIS) to Boeing and a contract for a software-support environment (SSE) to Lockheed. The TMIS will enforce consistent management and documentation, while the SSE will enforce consistent tool use, simulation techniques, and interfaces.

Management. The space agency has approved software-management policies intended to ensure top-level policy direction and common software policies. The policies address a standard software life-cycle model specifying the phases of software acquisition, associated reviews, documentation policies, and standard terminologies.

Key software development will occur at the Marshall Space Flight Center in Huntsville, Ala., the Johnson Space Center in Houston, the Goddard Space Flight Center in Greenbelt, Md., and the Lewis Research Center in Cleveland, Ohio. International partners — Canada, Japan, and the European Space Agency — will also develop secondary hardware and software systems.

NASA headquarters in Washington, D.C., is ultimately responsible for enforcing standards and integrating the system pieces (including simulators) supplied by the various sites. To help do this, software components, algorithms, and simulation packages will be kept in libraries available to all centers, ensuring consistency and trustworthiness as well as minimizing replication. However, NASA engineers must yet establish a methodology to ensure the consistency and quality of library components.

The Marshall team has begun planning and implementing a software-management approach to accomplish its assignment. While the project's scope is larger than previous NASA efforts, Marshall software managers are following techniques from current and past programs, including Skylab and the space shuttle.

Based on updated versions of the 1979 benchmark "MSFC Software Management and Development Requirements" document (MA-001-006-2H) and consistent with the overall software-management policy document, Marshall project managers have developed a software-management plan that establishes policies, procedures, and guidelines to be used during all development phases.

The Marshall team has begun planning for system integration and testing. It already has plans for verification and validation of software integration of the habitation, laboratory, and logistics modules. Because these elements will include logic-software packages provided by other centers, the Marshall software manager is developing plans to integrate and test software provided by other centers.

Design. The software will contain not only the unique flight-software packages required to operate the station but also many services that in the past were found in ground-based software.

The on-board and ground systems will rely on custom-built software, commercial off-the-shelf software, and firmware packages. Decisions have yet to be made about the systems' architectures, but several prototype architectures are being tested. Regardless of the configuration, the system will require a high degree of reliability while performing real-time and near-real-time functions and sustaining operations 24 hours a day.

Considering the station's potential 30-year life span and the rate at which technology is moving, keeping the space station software current presents a set of unique challenges for the systems manager. Initial planning must let capabilities under development and those not yet thought of be added without disrupting the station's operational capabilities. This is one of the biggest challenges NASA will face, and NASA must develop a methodology to handle this problem as the insertion of new technology over the next 30 years.

Expert systems may play a role in the space station software, but many questions remain about their trustworthiness. One approach under consideration is to implement expert systems in parallel to other systems (but off-line) and compare their decisions to those made by NASA engineers. If the expert systems make appropriate decisions, they could be made part of the actual system. The key question will be determining if these expert systems really work before using them to automate the station or parts of it.

Quality Assurance. The program's hierarchical management, standards enforcement, common work approaches, and use of library components are designed to ensure that defects are caught before they become critical.

Simulation at several levels will help verify quality. Packages will be tested independently, as related groups, as groups of groups, and ultimately as the entire system. This system simulation will most likely continue to be used after the space station itself is functioning so NASA can test changes, upgrades, and fixes before applying them to the real station. NASA has used this technique successfully since the Apollo missions to the Moon.

Wolfsberger is lead engineer for new software technology at the Marshall Information and Electronics Data Systems Laboratory. He has worked with NASA since the Apollo missions.

Correction

The September issue's report on software-engineering programs (pp. 88, 90) mistakenly said that Texas Christian University's software-engineering program no longer exists. The graduate program has run continuously since 1978, although it changed its name in 1981 from software engineering to software design and development, said James Comer, the university's Computer Science Dept. chairman. The program offers six courses to about 50 students, most of whom work for local industry, he said.
Software Paradise

Gentle ocean breezes, breathtaking sunsets, and the most exotic software projects you can imagine. That's what awaits qualified software engineers at Lockheed Missiles & Space Company in Sunnyvale.

We offer you a software paradise full of rare challenges and opportunities. Along with access to advanced computer systems, such as APOLLO, SYMBOLICS, VAX AND CDC CYBERS. Experienced software engineers are invited to explore the following career paths:

Software Requirements Analysis, Design and ADA Code Development
- Five or more years' software engineering experience
- ADA design and code development experience is preferred.

Simulation and Real-Time Code Design
- Experience with VAX/VMS, FORTRAN and PDP 11/70 Assembly Language is required.

Real-Time and Operating Systems Development
- Eight or more years' software engineering experience
- Knowledge of FORTRAN and VAX/VMS is required.

FORTRAN Software Development
- Four or more years' experience in a VAX/VMS environment
- Experience in real-time modeling, VMS Systems Management, FMS/Datatrieve, Database Management or graphics software is preferred.
- Experience with high-priority defense programs is strongly desired.

Take a walk on the wild side. Send your resume to Professional Staffing, Dept. 533HRLB, Lockheed Missiles & Space Company, P.O. Box 3504, Sunnyvale, CA 94088-3504. We are an equal opportunity affirmative action employer. U.S. citizenship is required.

Compsac Continued from p. 93

"We have various issues in trade, etc. that need to be resolved," NEC's Sekimoto acknowledged. But Japan is contributing substantially to software-development technology transfer, he said.

Information society. The role Japan has been playing domestically to begin transforming itself into an information society was a focus throughout the conference, held for the first time outside Chicago, partly to acknowledge Japan's increasingly major role in software research.

The Japanese government, through its Ministry of International Trade and Industry, has been especially forceful in its research efforts in productivity improvement and mass-use computing.

"Our purpose is to push for an information society," said MITI Vice Minister Shizuma Kojima at the keynote session. "[But] we cannot promote this policy just as a policy of Japan. We need to cooperate," he said.

MITI has three large programs under way to help accomplish the transformation of Japan: the ICOT fifth-generation language project, the Sigma software-development environment project, and the Iron project to apply artificial intelligence and microprocessors to everyday tasks for millions of users.

The Japanese programs share a belief that expert systems will play a key role in solving user-interface, development-tool, and system-management needs, at both the development sites and in the final systems. "We need to pursue intelligence" in machines so they help the user make decisions and handle routine needs automatically when possible, Sekimoto said.

Sekimoto foresees distributed systems with common protocols for data interchange and systems that use one database with several data representations. Such systems will let the various manufacturers' systems work together, keeping development costs low and letting the average user — the shopkeeper or butcher — use computer systems without much training or effort, he said. To accomplish this, software developers must think in terms of developing systems, not just software, he said. This integrated view will help make usable, everyday systems a reality, Sekimoto said.

In the short term, the best productivity booster appears to be increasing computing power each engineer has, several speakers said — although there was some dispute over whether PCs, workstations, minicomputers, or mainframes were the best host machines.